

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開平9-244940

(43) 公開日 平成9年(1997)9月19日

(51) Int.Cl. <sup>6</sup>	識別記号	庁内整理番号	F I	技術表示箇所
G 0 6 F 12/00	5 4 5		G 0 6 F 12/00	5 4 5 B
	5 3 5			5 3 5 Z
15/16	3 8 0		15/16	3 8 0 D

審査請求 未請求 請求項の数 7 O L (全 20 頁)

(21) 出願番号 特願平8-54398

(22) 出願日 平成8年(1996)3月12日

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 増岡 義政

東京都国分寺市東恋ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(72) 発明者 鍵政 豊彦

東京都国分寺市東恋ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(72) 発明者 北井 克佳

東京都国分寺市東恋ヶ窪1丁目280番地

株式会社日立製作所中央研究所内

(74) 代理人 弁理士 小川 勝男

最終頁に続く

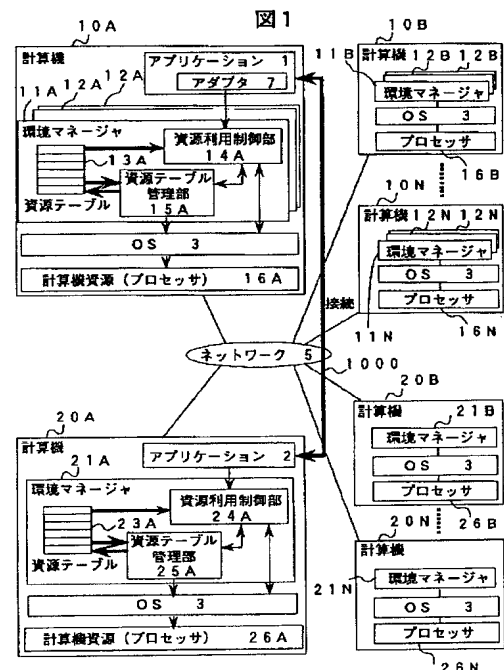
(54) 【発明の名称】 分散計算機資源の管理方法

(57) 【要約】

【課題】 アプリケーション1が別のアプリケーション2と通信を開始する際、アプリケーション1用の分散計算機資源をアプリケーション2から使用可能とする。

【解決手段】 アプリケーション対応に、そのアプリケーションが利用できる計算機資源を管理する資源テーブルを設け、あるアプリケーションが別のアプリケーションと接続、そして共同処理を行う際は、それぞれの資源テーブルを互いにコピーして両方の資源を管理する資源テーブルを作り、これを各アプリケーションが参照することで、各アプリケーションが両方の計算機資源を利用できるようにする。

【効果】 2つのアプリケーションが、それまで自分が利用できた分散計算機資源を互いに利用でき、より多くの分散計算機資源を有効利用でき、実行性能を高められる。



## 【特許請求の範囲】

【請求項1】 ネットワークにより相互接続された複数の計算機からなる分散計算機における分散計算機資源の管理方法であって、

アプリケーション対応に、そのアプリケーションが利用できる計算機資源を管理する資源テーブルを設け、あるアプリケーションが別のアプリケーションと共同処理を行う際は、それぞれの資源テーブルを互いにコピーして両方の資源を管理する資源テーブルを作成し、各アプリケーションが前記作成された資源テーブルを参照することを特徴とする分散計算機資源の管理方法。

【請求項2】 ネットワークにより相互接続された複数の計算機からなる分散計算機における分散計算機資源の管理方法であって、

各計算機上の少なくとも一つ以上のアプリケーションが利用できる計算機資源を管理する資源テーブルを各アプリケーション毎に設けるステップ、

計算機上の第1のアプリケーションが計算機上の第2のアプリケーションと通信を行おうとする際にそれを検出するステップ、

前記通信を行うことを検出した際、第1のアプリケーション用の資源テーブルをコピーし、第1の資源テーブルを作成するステップ、

前記通信を行うことを検出した際、第2のアプリケーション用の資源テーブルをコピーし、第2の資源テーブルを作成するステップ、

前記第1の資源テーブルに前記第2の資源テーブルの内容を追加するステップ、

前記第2の資源テーブルに前記第1の資源テーブルの内容を追加するステップ、

前記第1のアプリケーションは前記第2の資源テーブルの内容が追加された前記第1の資源テーブルを参照して通信を行うステップ、および前記第2のアプリケーションは前記第1の資源テーブルの内容が追加された前記第2の資源テーブルを参照して通信を行うステップを有することを特徴とする計算機資源の管理方法。

【請求項3】 ネットワークにより相互接続された複数の計算機からなり、各計算機上のアプリケーションが利用可能な計算機資源を示す情報をもとに実行される、分散計算機における分散計算機資源の管理方法であって、

アプリケーションが他のアプリケーションと通信を行おうとする時にそれを検出するステップと、

前記アプリケーションから前記他のアプリケーションへの通信の検出に応じ、前記他のアプリケーションに対応する他の資源マネージャに対して、前記アプリケーション自身の計算機資源を示す情報を送信するステップと、前記他の資源マネージャが、前記送信された計算機資源を示す情報の受信に応じて、前記受信した情報を自分の計算機資源の情報に追加するステップとを含んでいることを特徴とする分散計算機資源の管理方法。

【請求項4】 前記資源マネージャが、前記アプリケーションごとと独立して設けられていることを特徴とする請求項3に記載の分散計算機資源の管理方法。

【請求項5】 他のアプリケーションに計算機資源を示す情報を転送する際は、まずそのコピーを作るステップと、

アプリケーションは、前記計算機資源を示す情報を受信した際は、まず自アプリケーション用の計算機資源をコピーし、コピーした計算機資源を示す情報に受信した計算機資源を示す情報を追加するステップを有することを特徴とする請求項3に記載の分散計算機の管理方法。

【請求項6】 前記計算機資源を示す情報が、前記ネットワーク上に分散して存在する複数のプロセッサの負荷情報を含むことを特徴とする請求項3乃至5に記載の分散計算機資源の管理方法。

【請求項7】 前記アプリケーションと前記他のアプリケーションは、クライアントとサーバの関係にあることを特徴とする請求項3乃至6に記載の分散計算機の管理方法。

## 20 【発明の詳細な説明】

## 【0001】

【発明の属する技術分野】 本発明は、ネットワークを通信手段として利用する分散システムの管理方法に関し、特に、ネットワークに接続された複数の計算機に分散して存在している、プロセッサ、ファイルシステム、磁気ディスクなどを含む計算機資源（分散計算機資源）を、アプリケーションが利用できるようにする管理方法に係わるものである。

## 【0002】

30 【従来の技術】 近年のネットワークの普及、及び計算機の低価格化に伴い、多数の計算機がネットワークで接続されるようになった。そして、これらネットワークに接続された複数の計算機を利用してアプリケーションを実行する際、各計算機に分散して存在する計算機資源（分散計算機資源）をアプリケーションから有効利用する動きが強まっている。なお、個々の計算機資源（以下、適宜「資源」と省略する）としては、例えばプロセッサや、ファイルや、磁気ディスクの空き容量など、様々な種類を挙げることができる。

40 【0003】 このネットワークで接続された分散計算機環境を用いれば、例えば、1つの計算機上で実行されているアプリケーションが別のアプリケーションを新たに実行しようとする時、自アプリケーションが動作しているプロセッサとは別のプロセッサ上で別のアプリケーションを実行するようにすることにより、応答時間を短縮したり、データを新たに書き込むことのできる磁気ディスクの容量を増やしたり、取り扱うファイルの複製を分散配置することによりアプリケーション実行の信頼性を高めたりできるという効果がある。

50 【0004】 しかし、ネットワークに接続されたすべて

の計算機上の資源は、ネットワークに単に接続されているからといって、アプリケーションから利用可能となるわけではない。利用する手段が設けられていない、利用を許されていない、などの制約から、アプリケーションが利用できる分散計算機資源は、普通、前記全ての計算機上の資源のある部分集合となる。以下の説明では、この部分集合を「環境」と呼ぶことにする。本発明は、この環境を管理する手段に係わるものである。

【0005】アプリケーションが分散計算機資源を利用する際に重要な要素の1つとなる技術は、「透過性（transparency）」と呼ばれる性質を実現する技術である。この透過性は、アプリケーションが、前記の環境に含まれる個々の資源を、その資源が実際に存在するネットワーク上の計算機の位置を意識する必要なく、自分と同じ計算機上にある資源と同等に扱えるようになっていくという性質をいう。

【0006】この透過性を実現するため、これまでに多くの技術が提案されている。例えば、分散計算機資源としてファイルを取り扱うソフトウェアとして、Network File System (NFS) が良く知られており、この場合は、複数の計算機に分散したファイルシステムにアクセスできるようにするというのである。また、分散計算機資源としてプロセッサを取り扱う技術としては、Carnegie-Mellon大学のButlerシステムを挙げることができる。この場合は、一つのアプリケーションが新たにアプリケーションを実行する場合に、負荷の小さいプロセッサを選択してそこで実行できるようにするというのである。

【0007】NFSについては、文献「Andrew S. Tanenbaum, "Distributed Operating System", Prentice-Hall, pp. 272-278, 1995」で説明されている。また、Butlerシステムについては、文献「David A. Nichols, "Using Idle Workstations in a Shared Computing Environment", Proceedings of 11th Symposium on Operating Systems Principles, ACM, pp. 5-12, 1987」で述べられている。両者とも、アプリケーションが分散計算機資源を利用できるようにする方式の典型であるが、これらは、基本的に図2のような構造になっている。

【0008】図2は計算機100A上で実行されている1つ又はそれ以上のアプリケーション101が、資源マネージャ102Aおよび102Bにより、別の計算機100B内の資源108を、分散計算機資源の1つとして利用している様子を示している（なお、資源マネージャに関しては、NFSについてはNFSクライアントおよびNFSサーバという名称があり、Butlerシステ

ムについてはremおよびbutlerという名称があるが、ここではこれらを「資源マネージャ」と総称する）。アプリケーション101は、資源108が実際にはどの計算機内にあるかといったことは意識することなく、資源108の利用を要求する（104）。資源マネージャは各計算機ごとに1つずつ設けられており、計算機100Aの資源マネージャ102Aの資源利用制御部109は要求104を受けて、自分が保持している資源テーブル103を参照し（105）、資源108のある計算機100Bを見だし、計算機100B内の資源マネージャ102Bに、アプリケーション101からの要求をネットワーク5を経由して転送する（106）。ここで、資源テーブル103の参照から、資源マネージャ102Bへの要求の転送の間に、102Aおよび102Bとは別の資源マネージャとの通信を伴う場合もある。資源マネージャ102Bは、資源マネージャ102Aからの要求を受け、資源108の利用要求を発行し（107）、アプリケーション101の要求に応じてしかるべき応答を返す。

【0009】即ち、各計算機に資源マネージャ102A、102Bを設けておくことにより、アプリケーション101は、別計算機上の計算機資源をアクセスする機能を自ら準備する必要はなく、資源マネージャ102A、102Bを利用することによって、資源テーブル103に記載の分散計算機資源を容易に利用することができる。

【0010】以上のような、資源マネージャの存在により、アプリケーションが分散計算機資源を利用可能となっている分散プログラミング環境において、利用可能な計算機資源が何らかの理由で増加した場合を考える。一般に、このような場合、この新しく利用可能となった資源を、これまで資源マネージャが管理してきた資源に追加することができれば、分散計算機資源の有効利用によりアプリケーションが得られる効果を、より大きなものとすることができる。

【0011】この環境の拡張に関しては、例えば前記NFSでは、ある特権を持つユーザに限り、追加する計算機資源（ファイルシステム）を指定することにより、そのファイルシステムを、資源マネージャの管理対象として、資源テーブルに新たに加える手段が提供されている。また、Butlerシステムでは、ある計算機上で資源マネージャを起動すると、分散ファイルシステムによるファイル共有により、その計算機のプロセッサを利用するための情報を、他の資源マネージャの資源テーブルに追加する機能が提供されている。

【0012】一方、アプリケーションに目を転ずると、最近、広域ネットワークを介した通信により、別のアプリケーションと協調するアプリケーションが多くなっている。こうしたアプリケーション間の協調方式として一般に用いられているのは、クライアント・サーバモ

デルと呼ばれる方式である。クライアント・サーバモデルでは、ある機能を計算機サービスとして提供するアプリケーション（サーバ）がある計算機上で実行されており、その計算機サービスを利用しようとする別のアプリケーション（クライアント）は、広域ネットワークを経由してサーバに通信して自分の要求を伝える。サーバは、クライアントからの要求を受け付けて、然るべき処理を実行し、クライアントに応答を返す。

【0013】このクライアント・サーバモデルを含め、広域ネットワークを介して複数のアプリケーションが協調する際（このような環境を「分散プログラミング環境」という）の特徴は、それまで、プログラム上も、地理的にも、全く無関係であった複数のアプリケーションが、実行中に新規に通信を開始し、それ以後協調動作をするところにある。

【0014】

【発明が解決しようとする課題】従来技術の図2で示した分散計算機環境では、それぞれの計算機において、複数のアプリケーションが、1つの資源テーブルを有する1つの資源マネージャによって前記複数のアプリケーションを一括して管理していた。従って、これら複数のアプリケーションの内の一部の複数のアプリケーションが協調してある処理を実行しようとする場合、更に、他の計算機内のアプリケーションと協調させて処理を実行させる場合、資源テーブルからある特定のアプリケーション用の計算機資源を選択し、これを協調させる相手のアプリケーションの計算機資源とまとめる処理など、協調して処理を行う全てのアプリケーションに自由に使用させるには、その処理が複雑になっていた。

【0015】即ち、あるアプリケーションが利用可能な分散計算機資源をまとめて直ちに知る手段が無く、複数のアプリケーション用の分散計算機資源をまとめる効率的な方法が存在しなかった。

【0016】又、特別な権限を持った管理者が行っており、そのため、分散計算機資源の拡張は、アプリケーションが分散計算機資源の拡張が必要としたその時点では、動的に拡張することができなかった。そのため、例えば、分散計算機資源の拡張が実行されまでに多くの時間を要し、たとえ分散計算機資源の拡張が行われたとしても、アプリケーションのトータルな応答時間が短縮されるなどの効果を得ることは困難であった。

【0017】そこで、自分の利用可能な分散計算機資源を、まとめて相手のアプリケーションが利用できるようにしたり、逆に相手のアプリケーションが利用可能な分散計算機資源を、まとめて自分が使えるようにする手段が設けられていると、利用可能な分散計算機資源を効率的に、かつ大きく増やすことができ、アプリケーションの実行性能を高める上で、大きな効果が期待される。

【0018】本発明は、上記課題を解決するものであり、特に広域ネットワークに接続された計算機から構成

された大規模な分散プログラミング環境において、複数のアプリケーションが分散計算機資源を拡張利用できるようにする分散計算機資源の管理方式を提供する。また、利用可能な分散計算機資源を効率的に拡張できる分散計算機資源の管理方式を提供する。

【0019】

【課題を解決するための手段】アプリケーション対応に、そのアプリケーションが利用できる計算機資源を管理する資源テーブルを設け、あるアプリケーションが別のアプリケーションと接続、そして共同処理を行う際は、それぞれの資源テーブルを互いにコピーして両方の資源を管理する資源テーブルを作り、これを各アプリケーションが参照することで、各アプリケーションが両方の計算機資源を利用できるようにする。

【0020】具体的には、ネットワークにより相互接続された複数の計算機からなる分散計算機において、各計算機上の少なくとも一つ以上のアプリケーションが利用できる計算機資源を管理する資源テーブルを各アプリケーション毎に設けるステップ、計算機上の第1のアプリケーションが計算機上の第2のアプリケーションと通信を行おうとする際にそれを検出するステップ、前記通信を行うことを検出した際、第1のアプリケーション用の資源テーブルをコピーし、第1の資源テーブルを作成するステップ、前記通信を行うことを検出した際、第2のアプリケーション用の資源テーブルをコピーし、第2の資源テーブルを作成するステップ、前記第1の資源テーブルに前記第2の資源テーブルの内容を追加するステップ、前記第2の資源テーブルに前記第1の資源テーブルの内容を追加するステップ、前記第1のアプリケーションは前記第2の資源テーブルの内容が追加された前記第1の資源テーブルを参照して通信を行うステップ、および前記第2のアプリケーションは前記第1の資源テーブルの内容が追加された前記第2の資源テーブルを参照して通信を行うステップを設ける。

【0021】また、ネットワークにより相互接続された複数の計算機からなり、各計算機上のアプリケーションが利用可能な計算機資源を示す情報をもとに実行される、分散計算機において、アプリケーションが他のアプリケーションと通信を行おうとする時にそれを検出するステップと、前記アプリケーションから前記他のアプリケーションへの通信の検出に応じ、前記他のアプリケーションに対応する他の資源マネージャに対して、前記アプリケーション自身の計算機資源を示す情報を送信するステップと、前記他の資源マネージャが、前記送信された計算機資源を示す情報の受信に応じて、前記受信した情報を自分の計算機資源の情報に追加するステップとを設ける。

【0022】次に又、前記資源マネージャが、前記アプリケーションごと独立して設けられていることを特徴とする。次に又、他のアプリケーションに計算機資源を示

す情報を転送する際は、まずそのコピーを作るステップと、アプリケーションは、前記計算機資源を示す情報を受信した際は、まず自アプリケーション用の計算機資源をコピーし、コピーした計算機資源を示す情報に受信した計算機資源を示す情報を追加するステップを設ける。

#### 【0023】

【発明の実施の形態】以下、本発明の実施の一形態を、図面を参照しながら説明する。なお、簡単のため、本明細書中では「発明の実施の形態」を、単に「実施形態」と呼ぶことにする。

【0024】ここで、計算機資源とは、アプリケーションプログラムを実行するプロセッサや、プログラムやデータを格納したり取り出したりすることができる磁気ディスクやファイルシステム、などの総称である。なお、本実施形態では、計算機資源としてプロセッサを取り上げ、このプロセッサにアプリケーションを動的に割り付ける場合、即ち、アプリケーションのプロセッサへの負荷分散に適用した場合について、本発明の計算機資源管理方法を詳細に説明する。ただし、これは本発明が利用可能なプロセッサの管理に限定されることを意味するものではなく、以下の説明から理解されるように、磁気ディスクやその他の分散計算機資源の管理にも容易にも適用可能なものである。

【0025】以下、本実施形態を順序説明する。始めに、本実施形態において、アプリケーションが自分が利用可能な分散計算機資源、すなわち環境を利用する手段を説明する。その次に、本実施形態において本発明の環境を拡張する手段について、まずその動作の概略を示し、その次に、その構造および手続きを述べる。

【0026】まず、本発明を適用する計算機ネットワークについて説明する。

【0027】(1) 計算機、計算機資源、ネットワーク通信、オペレーティングシステム

図1は、複数の計算機がネットワークにより相互接続されている分散計算機システムを示している。この分散計算機システムが本発明を適用する対象となる。各計算機10A~10N、20A~20Nは、それぞれがプロセッサ16A~16N、26A~26Nを有する。

【0028】今、計算機10Aでは、オペレーティングシステム(OS)3の管理の下で、1つまたはそれ以上のアプリケーション(ユーザプロセス、タスク、スレッド、などとも呼ばれるが、本実施形態では、アプリケーションとして統一する)を、プロセッサ16Aを利用して実行させることができる。計算機10Aにおいて、実行中のアプリケーションは、OS3に「複製(fork)」「実行(exec)」要求を発行することにより自分自身のアプリケーションを新規に実行させたり、「終了」要求をOSに発行することにより自分自身の実行を終了したりすることができる(「複製」「実行」の両要求は、アプリケーションを新規に実行させるという

点では同等で、分散計算機資源の管理方法である本発明の実施形態を説明する限りにおいては、両者の相違は重要ではないので、本実施形態では適宜、総称して「起動」要求と呼ぶ)。以上のように、アプリケーションは、OS3に要求を発行することにより、計算機資源としてのプロセッサ16Aを利用可能である。

【0029】また、計算機10Aのアプリケーションは、相手のアプリケーションを指定してOS3に要求を発行することにより、ネットワーク5に接続された計算機10B~10Nおよび20A~20N上で動作している他のアプリケーションと、適当なデータを送受信し合うことができる。

【0030】上記プロセッサ10Aおよびプロセッサ16Aに関して述べたことは、計算機10B~10Nおよびプロセッサ16B~16Nについても同様である。

【0031】ネットワーク5を経由したアプリケーション間の通信においては、文献「W. Richard Stevens, "UNIX Network Programming", Prentice-Hall, 1991, pp. 171-196」に述べられているように、次のような手順がとられる。

【0032】ある他のアプリケーション(「サーバ」と呼ぶ)と通信しようとするアプリケーション(「クライアント」と呼ぶ)は、サーバを指定して、OS3に「接続要求(connect)」要求を発行する。サーバの指定は、サーバアプリケーションの実行されている計算機を特定するための識別番号(識別ID)と、その計算機内で、そのアプリケーションを特定するためにアプリケーション対応に設けられた識別番号(識別ID)の組み合わせ(「アドレス」)によって行なわれる。

【0033】以下、アドレスとして計算機の識別番号(識別ID)と識別番号(識別ID)の組として定義する。このアドレスは、同一計算機内のアプリケーション相互間の通信においては必要に応じて、計算機内で、そのアプリケーションを特定するためにアプリケーション対応に設けられた識別番号(識別ID)のみが使用される場合もある。

【0034】一方、サーバは、「接続受付(accept)」要求をOS3に発行し、クライアントからの接続要求を待つ。クライアントがこのサーバに対して「接続要求」要求を発行し、サーバがこの接続要求を受けると、クライアントとサーバとの間で接続(connection)が確立され、その旨がクライアントとサーバの両アプリケーションに通知される。これ以後、両アプリケーションはこの接続を通じて、適当なデータを相手と送受信することができる。

【0035】通信が終了した時は、クライアントあるいはサーバのどちらか一方が終了するか、その接続を指定してOS3に「閉鎖(close)」または「遮断(shutdown)」要求等をアプリケーションからOS

3に発行することにより、接続を切り離す。

【0036】以上述べた各要求を実行する機能を有するOS3としては、例えば、文献「B. Goodheart他、"The Magic Garden Explained: The Internals of UNIX System V Release 4", Prentice-Hall, 1994, pp. 20-65」に記載がある。しかしながら本発明は、前述のOSに限定して適用されるものでなく、他のOSに対しても容易に適用可能である。

【0037】(2)環境マネージャによる分散計算機資源の利用

本発明では、上記のネットワーク5を介した計算機システムに対して、以下説明する構造を持つ資源マネージャ(以下「環境マネージャ」と呼ぶ)を設ける。

【0038】図1において、各計算機10A~10Nには、それぞれ1つ以上の環境マネージャ11A~11Nおよび12A~12Nが設けられている。環境マネージャ11A~11Nは、アプリケーション1の利用可能な分散計算機資源の集合(すなわち環境)を管理しており、別のアプリケーションの環境は、環境マネージャ12A~12Nによって管理されている。即ち、環境マネージャはアプリケーション対応に設けられる。

【0039】環境マネージャ11Aは、アプリケーション1が利用可能な計算機資源を記録する資源テーブル13A、アプリケーションからの要求に応じて適切な計算機資源を資源テーブルを基に決定しアプリケーションへ報告する資源利用制御部14A、資源テーブルへの内容の追加、変更、削除など資源テーブルの管理を行う資源テーブル管理部15Aから構成されている。他の計算機内の環境マネージャ11B~11Nおよび12A~12Nの構造もこれと同じである。

【0040】尚、本実施形態においては、上述の環境マネージャ11Aの資源利用制御部14A、および資源テーブル管理部15Aは、それぞれOS3の管理下で実行されているアプリケーションの一つであるが、以下では資源利用制御部、資源テーブル管理部という呼称のみを用い、資源利用制御部14Aおよび資源テーブル管理部15Aを利用するアプリケーションなどと区別する。また、資源利用制御部14Aおよび資源テーブル管理部15Aは、前述のアドレスをそれぞれ1つづつ有しており、そのアドレスを指定することによってその機能をアクセスすることができる。具体的には、前述のStevensの文献等に記載されている方法により実現できる。本発明においては、単にOSが提供するアクセス機能を用いてこれらアプリケーション間の通信を行うというものであり、この点は発明の本質ではなく、これに関するこれ以上の説明は省略する。

【0041】資源利用制御部14Aは、アプリケーション1からの要求を、アダプタ7からの通知の形で受信

し、資源テーブル13Aの内容を読み出し、最適な環境マネージャ(本実施形態の負荷分散の場合は、負荷のもっとも小さい計算機上の環境マネージャ)を決定し、この決定した環境マネージャの資源利用制御部と通信する。この際、最適な環境マネージャが属する計算機が当該計算機以外だったらネットワーク5を介して通信する。そして、アプリケーション1が最適な分散計算機資源を利用できるようにする。

【0042】また、資源利用制御部14Aは、アプリケーションの実行や終了等によりプロセッサへの負荷が変化したり、新たな計算機の接続や計算機能力の向上やファイルの変更等などにより環境が拡張された時など、資源テーブル13Aの内容に変化が生じる時は、資源テーブル管理部15Aと通信し、資源テーブル管理部15Aに対して資源テーブル13Aの更新を要求する。

【0043】環境マネージャ11Aの資源テーブル管理部15Aは、当該環境マネージャ11A内の資源テーブル13Aのテーブル内容を管理するアプリケーションである。資源テーブル管理部15Aは、資源利用制御部14A、または環境マネージャ11A以外の環境マネージャの資源テーブル管理部と通信し、それらからの要求に応じて資源テーブル13Aの内容を更新し、また更新した旨を資源テーブル13Aに記載された他の環境マネージャの資源利用制御部に知らせる。この資源テーブル管理部15A~15Nの相互間の通信により、資源テーブル13Aの内容の変化は直ちに他の環境マネージャ11B~11Nの資源テーブル13A~13Nに反映される。

【0044】図3は、図1で環境マネージャ11A~11Nが存在している状態における、環境マネージャ11Aの資源テーブル13Aの内容を示す。この資源テーブル13Aは、資源利用制御部14Aおよび資源テーブル管理部15Aのいずれからも参照(読み出し)可能な、計算機10Aのメモリ上の領域である。ただし、資源テーブル13Aに対して書込みを行うことができるのは資源テーブル管理部15Aのみであり、資源利用制御部14Aは資源テーブル13Aに対して読み出しだけを行う。この資源テーブル13Aの内容により、アプリケーション1が利用可能な分散計算機資源、即ち、アプリケーション1が利用可能な一つの環境が定義される。

【0045】資源テーブル13Aは、計算機10A~10Nのプロセッサ16A~16Nのそれぞれのプロセッサにつき1つのエントリ130を成すテーブルである。この資源テーブル13Aには、各エントリに対応するプロセッサ16A~16Nに関して、プロセッサの性能、命令コード体系等の種別を示すプロセッサの種類132、現在のプロセッサの負荷133、及びそのプロセッサの存在する計算機の環境マネージャ11A~11Nへ他の計算機から通信を行う際の通信方法に関する情報131が格納されている。

【0046】ここで、負荷133としては、数値化されている値が用いられる。この負荷の数値化については、例えばNicol'sの文献等に記載されている方法を用いればよい。

【0047】また、環境マネージャ11A~11Nへの通信方法に関する情報131として、環境マネージャ11A~11Nのアドレスが格納される。具体的には、このアドレスは、資源利用制御部14A~14Nのアドレスと、資源テーブル管理部15A~15Nのアドレスから構成される。以下、このアドレスの組を「環境マネージャのアドレス」と呼ぶことにする。このアドレスには、前述のアドレスの定義で説明したように、環境マネージャ11B~11Nの存在する計算機の識別番号と、その計算機内での環境マネージャに結びつけられた識別番号が含まれている。

【0048】このように、アプリケーション1用の環境マネージャ11Aは、アプリケーション1に係する全ての環境マネージャに関するアドレスを資源テーブルに有するので、常に環境マネージャ11B~11Nと必要に応じて直ちに通信が可能である。尚、当該アプリケーション1と直接関係ない環境マネージャ12B~12Nとは、資源テーブルにその環境マネージャに関するアドレスを有しないので直ちに直接通信することはない。このことは、アプリケーション1の環境を管理する環境マネージャ11A~11Nは、別のアプリケーションの環境を管理している環境マネージャ12A~12Nとは独立して存在しているといえる。

【0049】図4は、図1において環境マネージャ11A~11Nが存在している状態で、環境マネージャ11Aが、アプリケーションからの要求に応じて、分散計算機資源をアプリケーションに提供する際の手続き、即ち、環境マネージャ11Aが分散計算機資源をアプリケーションに対して利用可能にする際の手続を示したものである。なお、以下で記述するREXEC、NEWLOADなどの通知は、環境マネージャ同士の通信に用いられるものであり、その形式については図6に示される。これら手続はプログラムの形で実現される。

【0050】資源利用制御部14Aは、次の手続き(a)~(c)を有する。

【0051】(a) アプリケーション1からの、当該アプリケーションを実行できる計算機資源の割り付けを要求する「起動」要求を検出すると(401)、資源テーブル13Aを参照して計算機10A~10Nのプロセッサ16A~16Nのうちから1つを選び出す。選択する基準は、例えば、負荷分散を目的とした環境マネージャであれば、当該アプリケーションを実行でき且つ負荷がもっとも少ないプロセッサを選択する。そして、資源利用制御部14Aは、その選択したプロセッサが属する計算機の環境マネージャに、ネットワーク5を経由して、指定されたアプリケーションをそのプロセッサ上で

実行するのに必要な情報を含むREXEC通知402を送信する。

【0052】(b) 他の計算機内の環境マネージャ11B~11Nの資源利用制御部14B~14NからREXEC通知403を受信すると、REXEC通知の中で指定されるアプリケーションを実行する要求、即ち、「起動」要求をOS3へ発行して、REXEC通知403のデータ203に指定されたアプリケーションを、自分の計算機10A上で実行する(アプリケーションの実行に関するより詳細な点については、前記Nicol'sの文献等に記載がある。)。なお、REXEC通知の符号に関し、単に送信側からみて402とし、これを受信する側から見て403としている。

【0053】(c) 手続き(a)または手続き(b)によりアプリケーションが新たに実行されるとプロセッサに負荷の変化が生じるが、この負荷の変化を資源テーブルに反映させる必要がある。これを実現するため、前記(a)または(b)によるアプリケーションの新たな実行により、計算機10A~10Nのプロセッサ16A~16Nの1つの負荷に変化が生じたことを通知するため、資源テーブル管理部15AにNEWLOAD通知404を送信する。

【0054】資源テーブル管理部15Aは、次の手続き(d)および(e)を有する。

【0055】(d) 手続き(c)による資源利用制御部14AからのNEWLOAD通知404の受信、あるいは資源テーブル管理部からのOS3への定期的な負荷値の問い合わせ要求の発行により、自分の計算機10Aのプロセッサ4の負荷の変化を検出する(例えば前記Nicol'sの文献に記載の方法により)と、資源テーブル13Aの、そのプロセッサに対応するエントリ130の内容を更新する。更に、資源テーブル13Aの各エントリ130を参照し、そのエントリに記載された他の環境マネージャ11B~11Nの資源テーブル管理部15B~15Nに、当該計算機10Aのプロセッサの負荷が変化したことを知らせるSETLOAD通知405を送信する。

【0056】(e) 手続き(d)による他の計算機の環境マネージャ11B~11Nの資源テーブル管理部15B~15NからのSETLOAD通知406を受信し、資源テーブル13A内の、対応するエントリを受信したSETLOAD通知の内容に従って更新する。

【0057】以上説明した手続きを、環境マネージャ11A~11Nは持っている。そのため、計算機10A上で実行中のアプリケーション1は、環境マネージャ11Aにアプリケーションを実行する計算機資源を要求することで、計算機10A~10Nのプロセッサ16A~16Nを利用可能となっている。具体的には、アプリケーション1が新たなアプリケーションを起動しようとする場合、アプリケーション1は環境マネージャ11Aに対

して、新たに起動したいアプリケーションを指定して要求を発行する。アプリケーション1からの要求を受けた環境マネージャ11Aの資源利用制御部14Aは、手続き(a)および手続き(b)を実行することにより、必要に応じてプロセッサ16A以外のプロセッサ16B～16Nの内の1つで新たなアプリケーションを実行させる。なお、資源利用制御部14Aは、例えばプロセッサの負荷値が自分自身が属する計算機10Aがもっとも小さいならば、自分自身が属する計算機10Aに新たなアプリケーションを割り付けても一向に問題ない。

【0058】また、資源利用制御部14Aの手続き(c)、資源テーブル管理部15Aの手続き(d)および手続き(e)により、資源テーブル13Aにはプロセッサ16A～16Nの負荷が保持される。このため、(a)において負荷の最も軽い計算機を選択することにより、資源テーブル13Aに記載されたプロセッサ16A～16Nに負荷が分散され、アプリケーション1の実行速度を高めることが可能となる。

【0059】なお、実際には、資源利用制御部14Aと資源テーブル管理部15Aが資源テーブル13Aを参照したり、両者が互いに通信したりするのを実現するのは、「W. Richard Stevens, "UNIX Network Programming", Prentice-Hall, 1991, pp. 87-170」に詳細に述べられている方法等に基づき、OS3に然るべき要求を発行することにより行なわれる。本実施形態では簡単のために、資源テーブル13Aの参照や資源利用制御部14Aと資源テーブル管理部15Aとの間の通信におけるOS3の介在についての説明を省略している。

【0060】また、ネットワーク5を介した環境マネージャ間の通信は、前記Stevensの文献(pp. 171-196)、および前記Goodheart他の文献に記載された方法等に基づき、OS3に然るべき要求を発行することにより行なわれる。本実施形態では簡単のために、環境マネージャ間の通信におけるOS3の介在についての説明を省略している。

【0061】なお、図1において、計算機20A～20N上の環境マネージャ21A～21N、資源テーブル23A、資源利用制御部24A、資源テーブル管理部25Aは、アプリケーション2からの要求を受けて、プロセッサ26A～26Nに負荷を分散するための機能であること以外は、その構造、機能とも、それぞれ環境マネージャ11A～11N、資源テーブル13A、資源利用制御部14A、資源テーブル管理部15Aと同じである。

【0062】本発明は、以上説明した図1に示す計算機ネットワークにおいて、それぞれ独立して実行されていたアプリケーション1とアプリケーション2が、新たに通信を開始した(接続した)時(1000)に、それぞれのアプリケーションがそれまで利用可能であった分散

計算機資源を通信相手のアプリケーションにも利用可能とすることにより、環境を効率的に拡張する方法を提供する。

【0063】このアプリケーション1とアプリケーション2との接続は、それぞれOS3にこれらアプリケーションの接続を依頼する要求を発行することにより行われるが、この際に、本実施の形態では、環境マネージャ11Aおよび21Aが、この要求を検出し、両者のアプリケーションが通信を開始する前に、互いの資源テーブル13Aおよび23Aの内容を互いに送信し合い、受信した内容をそれぞれ自分の資源テーブルに追加する。これにより、本発明を適用した分散計算機においては、アプリケーション1とアプリケーション2とを接続されて処理を行う際には、新たに追加された資源テーブルの内容を参照することによって、通信相手が使用していた分散計算機資源を自分の分散計算機資源として利用可能となる。

【0064】次に、OSへの要求の検出について説明する。

【0065】(3) OSへの要求の検出

図5は、本実施形態における、アプリケーション1からOS3への要求を、環境マネージャ11Aの資源利用制御部14Aにおいて検出する機能を詳細に示す。図5において、アプリケーション1のプログラム中にはアダプタ7(ソフトウェア的に実現されるアダプタである)が挿入されている。

【0066】アダプタ7内のプログラムは、アプリケーションがOS3への要求を発行する時に、アプリケーションがOS3へ発行する要求の内、プログラム実行に関する要求、および、他のアプリケーションとの接続(協調)動作を指定する要求など、資源テーブルの内容に影響を与える要求を検出する。この検出は、例えば、アプリケーションからOSへの要求が関数名で行われることから、この関数名を調べることによって行われる。なお、これら以外の要求に対しては、そのままOS3へ要求を転送する。

【0067】要求を検出したアダプタは、検出された要求をアダプタ内に一時的に保持すると共に、当該要求を発行したアプリケーションに対応する環境マネージャ11Aの資源利用制御部14Aに対して、例えば、前記Stevensの文献(pp. 87-170)に記載の手段に基づき、SYSCALL通知501を送信し、アプリケーション1がOS3に(例えば「接続要求」などの)要求を発行した旨を通知する。

【0068】本実施形態では、アダプタ7が資源利用制御部14Aに対して前記のSYSCALL通知を送った後、資源利用制御部14AからのREADY通知を待つ。この期間、アプリケーション1が発行した要求はアダプタ7内で停止し、資源利用制御部14AからREADY通知502を受信した時に、再開される。



【0069】尚、アプリケーション1のプログラムへのアダプタ7は、アプリケーション1に最初から組み込む形での実現、また、アプリケーション1を起動した時に、自動的に挿入されるようにする形での実現、いずれでも可能である。いずれにしても、アプリケーション1のプログラムは、アダプタ7が挿入されていることを意識する必要がなく、したがって本実施形態においてアプリケーションの開発や利用が著しく困難となることはない。

#### 【0070】(4) 通知

図6に、本実施形態において、環境マネージャ間、アダプタと環境マネージャ、などの間の通信に用いられる通知200の形式を示す。前記手続き(a)～手続き

(e)および以下説明する手続き(f)～(r)で用いられる通信は、すべてこの形式を持つ。これらの通知を宛先アプリケーションを計算機内アドレスにより指定してOSに発行すると、OSはこれらの通知を宛先のアプリケーションへ送る。

【0071】通知は、その通知の種類を示す識別番号201と、データ長202、およびデータ203から構成される。識別番号201およびデータ長202は固定長であり、それぞれ0以上の整数が格納される。データ203は可変長であり、その形式は通知の種類(識別番号)によって異なる。図14は、本実施形態における通知の種類と、対応する識別番号201およびデータ203の内容を示す。データ長202には、各々の通知のデータ203のデータ長が格納される。

【0072】図14には、まだ説明していない通知もあるが、ここで各通知200の意味を説明する。

【0073】SETLOAD通知は、計算機10A～10Nのプロセッサ16A～16Nの1つの負荷に変化が生じたことを通知するために用いられる。データ203の部分には、負荷が変化したプロセッサに対応するエントリ130が格納されている。このSETLOAD通知は、計算機10A～10N上の中で、負荷の変化したプロセッサ上にある環境マネージャの資源テーブル管理部において、上述の手続き(d)により送信される。また、SETLOAD通知は、それ以外の計算機の資源テーブル管理部において受信され、手続き(e)によって処理される。

【0074】NEWLOAD通知は、SETLOAD通知と同様に、計算機10A～10Nのプロセッサ16A～16Nの1つの負荷に変化が生じたことを通知するために用いられる。NEWLOAD通知のデータ203には、負荷の変化したプロセッサに対応するエントリ130が格納されている。SETLOAD通知とは異なる点は、同一の計算機内の資源利用制御部と資源テーブル管理部との間の通信に使用し、ネットワーク5を経由した通信には用いない点である。このNEWLOAD通知の通信には2通りの経路がある。1つめは、資源利用制御

部14Aが、すでに述べた手続き(c)によってNEWLOAD通知を送信し、送信されたNEWLOAD通知を資源テーブル管理部15Aが受信する経路である。その後、NEWLOAD通知を受信した資源テーブル管理部15Aは、手続き(d)を呼び出す。もう1つは、後述する環境マネージャ11Aのコピーの後、資源テーブル管理部15Aが、手続き(h)によってNEWLOAD通知を送信し、新規に起動された、環境マネージャ11Aに従属する環境マネージャ31Aの資源テーブル管理部35Aの手続き(h2)がこのNEWLOAD通知を受信する経路である。後者の経路については、後に図8を用いて説明する。

【0075】FUSION通知は、データ203に資源テーブル13Aの内容が格納される。後に図9を用いて説明するように、FUSION通知は、異なるアプリケーション間で通信を開始する時に、それぞれのアプリケーションに対応する環境マネージャ31Aの資源利用制御部34Aが、手続き(i2)により互いの資源テーブル13Aの内容を交換するために用いられる。資源利用制御部34Aが受信したFUSION通知は、さらに資源テーブル管理部35Aに送信される。FUSION通知を受信した資源テーブル管理部35Aは、手続き

(k)により、FUSION通知のデータ203を、資源テーブル33Aの基からある13Aに対する拡張分53Aとして、資源テーブル33Aに追加する。

【0076】SYSCALL通知は、アプリケーション1がOS3への要求を発行した際に、アプリケーション1に対応する環境マネージャ11Aに、アプリケーション1がOS3に対して要求を発行したことを通知するために用いられる。すでに説明したように、SYSCALL通知は、OS3に対する要求を検出した時にアダプタ7によって資源利用制御部14Aに送信され、アプリケーション1に対応する環境マネージャ11Aの資源利用制御部14Aにより受信される。SYSCALL通知のデータ203には、アプリケーション1がOS3に対して発行した要求の種類と、その要求のパラメータが格納される。要求の種類は、0以上の整数で表され、OS3によって定められている。SYSCALL通知は、ネットワーク5を経由して別の計算機に送られることはない。従って、計算機によって異なるOS3が用いられ、アプリケーションからの要求の種類に対して、それぞれ異なる番号付けがされている場合でも、SYSCALL通知の送受信において送信側と受信側との間で、アプリケーションが発行した要求の種類につき誤りが生じることはない。

【0077】SUBEMS通知は、後で図7を用いて説明する、環境マネージャ11Aをコピーする際に、新しい環境マネージャ31Aのアドレスを知らせるために用いられる。SUBEMS通知のデータ203には、環境マネージャ11Aのコピーによって作成された、環境マ

ネージャ11Aに従属する環境マネージャ31Aのアドレス131が格納される。SUBEMS通知は、環境マネージャ31Aの資源利用制御部34Aにおける手続き(f2)から、環境マネージャ11Aの資源テーブル管理部15Aおよびアダプタ7に送信される。SUBEMS通知を受信した資源テーブル管理部15Aは、管理テーブル18Aに受信したSUBEMS通知のデータ203を格納することにより、環境マネージャ11Aに従属する環境マネージャ31Aのアドレスを保持する。また、SUBEMS通知を受信したアダプタ7は、それ以後、受信したSUBEMS通知のデータ203に記載されている資源利用制御部34Aのアドレスに、SYSCALL通知を送信する。

【0078】SUBEME通知は、異なるアプリケーションの間での通信の終了、あるいはアプリケーション自体の実行終了に伴い、そのアプリケーションに対応する環境マネージャ31A（環境マネージャ11Aに従属する）の実行を終了する際に、環境マネージャ31Aの実行の終了を知らせるために、後述する手続き(m2)により、環境マネージャ31Aの資源利用制御部34Aから、環境マネージャ11Aの資源テーブル管理部15A、およびアプリケーション1のアダプタ7へ送信される。このSUBEME通知のデータ203には、環境マネージャ11Aのアドレス131が格納される。

【0079】SUBEME通知を受信した資源テーブル管理部15Aは、管理テーブル18Aから、送信者である資源利用制御部34Aのアドレスに対応するエントリを削除する(OS3の機能により、SUBEME通知の受信の際に、資源テーブル管理部15Aは送信者のアドレスを知ることができる)。また、アダプタ7は、SUBEME通知を受信したら、それ以後は、SUBEME通知のデータ203に記載されている環境マネージャ11Aの資源利用制御部14Aへ、すでに説明したSYSCALL通知を送信する。

【0080】READY通知は、アプリケーション1に対応する環境マネージャ11Aの資源利用制御部14Aから、アプリケーション1のアダプタ7へ送られる通知であり、アダプタ7に対し、アプリケーション1に対する分散環境資源の準備ができたことを知らせると共に、停止されていたアプリケーション1の再開を要求する。

【0081】既に述べたように、アプリケーション1は、OS3へ要求を発行すると、アダプタ7に制御が移り、環境マネージャ11Aと同期を取るために実行を停止させられている。アダプタ7はREADY通知を受信したら、制御をアプリケーション1に返し、制御を移されたアプリケーションは、中断していた（停止していた）アプリケーション1の実行を再開する。なお、READY通知はデータ203を持たない。

【0082】EXTEND通知は、環境の設定を行う時（後で図13を用いて説明する）に用いられる。EXT

END通知による通信には、4通りの経路がある。

【0083】1つ目は、アプリケーション1のアダプタ7から環境マネージャ11Aの資源利用制御部14Aへの経路であり、前述のSYSCALL通知の代わりに送信される。この時、EXTEND通知のデータ203には、環境マネージャ14Aの資源テーブル13Aに新たに追加しようとする環境マネージャ61Aのアドレスが格納される。2つ目は、環境マネージャ11A内における資源利用制御部14Aと環境マネージャ61A内における資源利用制御部64Aとの間の通信経路である。3つ目は、環境マネージャ11A内における資源利用制御部14Aと資源テーブル管理部15Aとの間の通信経路である。4つ目は、環境マネージャの資源テーブル管理部15A～15N相互間の通信経路である。2つ目から4つ目までの場合は、EXTEND通知のデータ203には、環境マネージャの資源テーブルの内容が格納される。

【0084】環境の設定を行う時、アダプタ7は、SYSCALL通知ではなくEXTEND通知を資源利用制御部14Aに送信する。EXTEND通知を受信した資源利用制御部14Aは、更にEXTEND通知を用いて、資源利用制御部64Aと互いの資源テーブルを交換し、自分の資源テーブルに追加する。

【0085】またEXTEND通知は、資源利用制御部14Aが手続き(n)によってEXTEND通知を受信した後、資源テーブル13Aを管理する資源テーブル管理部15Aに、EXTEND通知で受信した内容を知照するのに使用される。この資源利用制御部14Aからの通知を受信した資源テーブル管理部15Aは、手続き(p)を呼び出し、データ203の内容を資源テーブル13Aに追加して、さらに環境マネージャ11B～11Nの資源テーブル管理部15B～15NにEXTEND通知を送信する。資源テーブル管理部15AからのEXTEND通知を受信した資源テーブル管理部15B～15Nでは、それぞれ手続き(q)を呼び出し、受信したEXTEND通知のデータ203の内容をそれぞれの資源テーブル13B～13Nに追加する。

【0086】REXEC通知は、異なる計算機でアプリケーションを新規に実行する場合に、環境マネージャ間で通信される。REXEC通知のデータ203には、実行すべきアプリケーションが格納されている。ここで、実行すべきアプリケーションとしては、実行可能なプログラムを格納しているファイルのファイル名でもよいし、あるいはプログラムそのものであってもよい。REXEC通知は、資源利用制御部14Aにおいて、手続き(a)によって、異なる計算機上の環境マネージャの資源利用制御部に対して送信される。REXEC通知を受信した資源利用制御部14Aは、手続き(b)を呼び出し、REXEC通知のデータ203によって（前述の通り）指定されたアプリケーションを、自分が実行されて

いる計算機10Aにて起動する。

【0087】すでに述べたように、通知はアダプタ7と資源利用制御部14Aとの間で交換される。この際、アダプタ7が資源利用制御部14Aの手続き(a)に通知200を直接送るわけではなく、具体的には次のステップによる。図には示さないが、資源利用制御部14Aには、通知200の種類にかかわらず通知200を受信するステップと、通知200の識別番号201およびデータ203の内容を調べ、必要な手続きを呼び出す(例えば、識別番号が4(SYSCALL)であり、データ203の内容に記載されたOS3への要求の識別番号が「起動」要求に対応していれば、手続き(a)を呼び出す)ステップが存在する。本実施形態では、識別番号201、データ203の内容、および送信元のアドレス(OS3の機能により取得可能)により、各通知の意味は識別可能であり、例えば資源利用制御部14AがSYSCALL通知を受信した時に誤って手続き(b)を呼び出すようなことはない。また、資源テーブル管理部15Aその他でも、これは同じである。

#### 【0088】(5) 環境の設定

本実施形態のこれまでの記述では、図1の計算機10A~10N(20A~20Nについても同様)において環境マネージャ11A~11Nがすでに動作しており、さらにそれぞれの環境マネージャの資源テーブル13A~13Nには環境マネージャ11A~11Nに対応するエントリがすでに存在している(すなわち1つの環境がすでに設定されている)としてきた。しかしながら、本来は、計算機10A~10Nがお互いに無関係であった状態から、こうした状態まで到達する手段、すなわち1つの環境を設定する手段が必要である。また、例えば新しい計算機をネットワーク5に接続した場合などのように、後述するアプリケーション間の通信によらず、アプリケーション1が利用可能な計算機資源を増やす手段があると、本発明の環境マネージャはより有用となる。本実施形態では、これらを次のように実現する。

【0089】図13は、図1における1つ以上の計算機10A~10N上で環境マネージャ11A~11Nが動作している状態において、環境マネージャ11Aにより、計算機60Aで動作している環境マネージャ61Aを追加する場合の手続きを示す図である。まず、アプリケーション1のアダプタ7は、EXTEND通知1201を環境マネージャ11Aの資源利用制御部14Aに送信する。ここで、EXTEND通知1201のデータ203には、追加しようとする環境マネージャ61Aのアドレスが格納されている(このアドレスを知る手段については後述する)。

【0090】ここで、資源利用制御部14Aに次の手続きを設ける。

【0091】(n) アプリケーション1のアダプタ7から拡張しようとする環境マネージャ61Aのアドレス

をデータとして含むEXTEND通知1201を受信したら、資源テーブル13Aの内容を読みだす。そして、資源テーブル13Aから読み出した内容をデータ203として格納したEXTEND通知1202を、環境マネージャ61Aの資源利用制御部64Aに送信する。その後、環境マネージャ61Aの資源利用制御部64Aから、送信したEXTEND通知の応答となるEXTEND通知1202を受信するまで待つ。応答であるEXTEND通知を受信したら、そのEXTEND通知を資源テーブル管理部15Aに送信する(1203)。

【0092】また、資源利用制御部64Aに、次の手続きを設ける。

【0093】(p) 資源利用制御部14AからのEXTEND通知を受信したら、資源テーブル63Aの内容を読みだし、それをデータ203として格納したEXTEND通知1202を、資源利用制御部14Aに応答として返送する。次に、受信したEXTEND通知を、資源テーブル管理部65Aに送信する。

【0094】資源テーブル管理部15Aおよび65Aには、次の手続き(q)(r)を設ける。資源テーブル管理部15Aの場合について記載するが、資源テーブル管理部65Aの場合でも、環境マネージャ61A内の処理となる以外は同じである。

【0095】(q) 資源利用制御部14AからEXTEND通知1203を受信したら、そのEXTEND通知を資源テーブル13Aに記載の環境マネージャの資源テーブル管理部15B~15Nに更に送信する(1204)。また、受信したEXTEND通知のデータ203の内容を資源テーブル13Aに追加する。

【0096】(r) 資源テーブル管理部15B~15NからEXTEND通知1206を受信したら、そのデータ203の内容を資源テーブル13Aに追加する。

【0097】以上の手続きにより、前述の環境の設定が可能となる。すなわち、まず計算機10A~10Nが無関係であったとして、それぞれの計算機でOS3により環境マネージャ11A~11Nを起動する。この時点では各環境マネージャの資源テーブル13A~13Nは自分自身に対応する1つのエントリ130しかない。ここで、ユーザは環境の設定のための特別なアプリケーション(ユーティリティ・プログラム)を環境マネージャ11Aのアプリケーション1として起動し、環境マネージャ11B~11Nのアドレスをそのアプリケーションに1つ1つ入力する。アドレスを入力されたアプリケーション1はアダプタ7を通じてEXTEND通知を資源利用制御部14Aに送信し、上記述べた手続きにより資源テーブル13A(およびすでに資源テーブル13Aに対応するエントリが存在する環境マネージャの資源テーブル)に1つづつエントリが追加されていき、最終的に図1の環境マネージャ11A~11Nの状態が実現される。

【0098】ユーザが環境マネージャ11B～11Nのアドレスを知る方法は様々であるが、例えば環境マネージャを起動した時に画面にアドレスを表示するなどの方法が考えられる。また、上で述べたユーザの行動を真似るプログラムを作るのも容易であり、それを用いることもできる。いずれにしても、この環境の設定は1度だけ行えばよく、何かアプリケーションを起動しようとするたびに実行する必要はない。

【0099】また、上で述べた新規に計算機を購入した場合なども、前記の環境の設定と同じ要領で、その計算機をアプリケーションが利用可能な計算機資源に追加することができる。本実施形態の方法によれば、この計算機資源の追加は、アプリケーション（ユーティリティ・プログラムではない、通常のアプリケーション）の実行中に行うことができ、追加された時点から、アプリケーションは新しい計算機資源を利用することができる。この間、一度アプリケーションを終了させる必要はない。

【0100】（6）アプリケーション間の通信開始を契機とする環境の拡張

アプリケーション間の通信開始時には、アプリケーション1が異なるアプリケーション2との間での通信開始

（接続）1000を契機として、お互いの環境を相互に利用できるようにする環境の拡張を行う。各々のアプリケーションに対応する環境マネージャ11Aおよび21Aが、アプリケーション1及びアプリケーション2がOS3へ発行する要求を検出し、両者のアプリケーションが通信を開始する前に、互いの資源テーブル13Aおよび23Aの内容をお互いに送信し合い、受信した内容をそれぞれ自分の資源テーブルに追加することにより、環境の拡張を実現する。以下本実施形態における環境の拡張方法について具体的に説明するが、まず動作の概略を示し、その後で各部分の動作を詳細に述べる。

【0101】図10および図11は、本実施形態を適用する際の留意点を示した図である。ここで、図10は第1の状態図とし、図11は図10の状態の後の第2の状態図とする。

【0102】図10では、合計4台の計算機10A、10B、20A、20Bが、ネットワーク5により相互接続されている。なお、図示はしていないが、図10および図12では、計算機資源（プロセッサ）およびOSは図1と同様に存在している。計算機10Aのアプリケーション1は、環境マネージャ11Aおよび11Bを通じて、2台の計算機10Aおよび10Bの計算機資源（プロセッサ）を利用可能であるとする。計算機20Aのアプリケーション2および計算機20Bのアプリケーション2Bは、環境マネージャ21Aおよび21Bを通じて、2台の計算機20Aおよび20Bの計算機資源（プロセッサ）を利用できる。但し、アプリケーション2とアプリケーション2Bは、お互いに無関係に実行されているアプリケーションであるとする。更に、例えば計算

機の新規追加や、既存の計算機の負荷の変化により、環境マネージャ21Aの資源テーブルの内容に変更が生じた場合、その変更は（すでに手続き（c）の説明、および環境の設定の項にて述べた方法により）直ちに環境マネージャ21Bに反映される（1011）。

【0103】図11は、本実施形態での環境の拡張の実現法の1つを示す、図10後の第2の状態図である。図10におけるアプリケーション1とアプリケーション2とを接続（1000）して図11の状態にする場合、各々のアプリケーションに対応する環境マネージャ11Aおよび21Aが、両者のアプリケーションが通信を開始する前に、互いの資源テーブルの内容を交換し（1001）、受信した内容をそれぞれ自分の資源テーブルに追加する。この結果、アプリケーション1とアプリケーション2は、互いの資源を共有することができ、両アプリケーションとも4台の計算機10A、10B、20A、20Bのプロセッサを利用できるようになる。即ち、環境の拡張が実現される。

【0104】しかし、環境マネージャ21Aの資源テーブルの変化に応じて、環境マネージャ21Bの資源テーブルも変化すると、アプリケーション2Bも上記4台の計算機を利用できるようになってしまう。即ち、アプリケーション2Bがアプリケーション2と無関係に実行されている場合、図11の方法により環境を拡張し、アプリケーション2Bが計算機10A、10Bを使えるようになるのは好ましくない。

【0105】図12は、図10及び図11で示した留意点を考慮した、本願の発明を説明する図である。本発明では、環境マネージャをコピーする機能を設けている。アプリケーション1とアプリケーション2との間の接続1000を契機に、まず環境マネージャ11A、21Aがコピーされ、それぞれの環境マネージャに従属する環境マネージャ31Aおよび41Aが作成される。図12は、計算機10Aに環境マネージャ11Aとコピーされた環境マネージャ31Aがあり、計算機20Aに環境マネージャ21Aとコピーされた環境マネージャ41Aがある所を示す。ここで、環境マネージャ31Aの資源テーブルの内容は環境マネージャ11Aの資源テーブルの内容と同じであり、環境マネージャ41Aの資源テーブルの内容は環境マネージャ21Aの資源テーブルの内容と同じである。そして、環境マネージャ31Aおよび41Aの作成後、環境マネージャ31Aと環境マネージャ41Aとの間で資源テーブルの交換1001が行われ、アプリケーション1および2の環境が拡張される。この拡張された環境を用いて、アプリケーション1とアプリケーション2とが実行される。即ち、通信を開始したアプリケーションとは無関係なアプリケーション2Bの環境に影響を与えずに、互いに通信するアプリケーション1と2のみが環境を拡張することができる。

【0106】なお、コピーの前に存在した環境マネージャ

ャ 11 A、11 B、21 A、21 B の資源テーブルの内容は変化せず、従ってアプリケーション 2 B の利用可能な計算機資源は変化しない。即ち、上記環境マネージャのコピーを行ない、このコピーされた環境マネージャ間で環境を拡張するので、他のアプリケーション 2 B の環境には影響を与えないで済む。

【0107】また、アプリケーション 1 とアプリケーション 2 が再び無関係となった時は、それぞれのアプリケーションの環境は、上記方法により拡張する前の環境に戻るのが望ましい。本実施形態では、アプリケーション 1 と 2 との通信の終了を、環境マネージャ 31 A および 41 A がそれぞれ検出し、環境マネージャの実行を終了させる。このようにしてアプリケーション 1 と 2 は、拡張された環境を削除し、その後の処理を、それぞれ環境マネージャ 11 A と 21 A を通して分散計算機資源を利用しつつ行う。このように、本実施形態では、アプリケーションの環境を拡張する前の環境に容易に戻すことができる。

【0108】以下、これまで説明してきた本実施形態における環境の拡張方法につき、環境マネージャのコピーの方法、アプリケーション間の通信を契機とした環境の拡張方法、およびアプリケーション間の通信の終了に際して、環境を拡張前に復旧する方法の順で、各部の動作を詳細に説明する。

#### 【0109】(7) 環境マネージャのコピー

図 7 および図 8 は、図 1 で示した環境マネージャ 11 A ~ 11 N、および環境マネージャ 21 A ~ 21 N が存在する状態において、環境マネージャ 11 A が自分のコピーを作成する機能を実現する手段を示している。このコピーを作成する機能は、後述する資源テーブルを拡張する手段において、環境マネージャ 11 A に従属する、環境マネージャ 11 A のコピーである環境マネージャ 31 A を作成するのに用いられる。

【0110】環境マネージャ 31 A の作成に関し、環境マネージャ 11 A 内の資源利用制御部 14 A に、次の手続きを設ける。

【0111】(f) 資源利用制御部 14 A は、OS 3 に「複製」要求を発行することで自分自身のコピーである資源利用制御部 34 A を作成し、この資源利用制御部 34 A を実行する (701)。

【0112】この新たに作成された資源利用制御部 34 A は、「複製」要求によって作成されたものであるから、基本的には、手続き (a)、手続き (b)、手続き (c)、その他の資源利用制御部 14 A が有する手続きを有する。更に、資源利用制御部 14 A が利用できる資源テーブル 13 A を、資源利用制御部 34 A から参照する機能 (手続き) を有する。また、資源利用制御部 34 A は、資源利用制御部 14 A と同様に、資源テーブル管理部 15 A、アダプタ 7 に対して通知を行なう機能を有する。

【0113】(f2) 資源利用制御部 34 A は、資源テーブル 33 A のための主記憶領域を確保する (703)。次に、資源テーブル管理部 35 A を作成し、実行する「複製」要求を OS 3 に発行する (704)。その後、資源テーブル管理部 15 A に対し SUBEMS 通知 705 を送信し、新しく資源テーブル管理部 35 A が作成されたことを通知する。さらに、アプリケーション 1 のプログラム内に挿入された前記アダプタ 7 に SUBEMS 通知 702 を送信する。この SUBEMS 通知 702 は、当該 SUBEMS 通知 702 を受信した後はアプリケーション 1 において検出した OS 3 への要求を資源利用制御部 34 A に送るようアダプタに要求する。

【0114】環境マネージャ 11 A の資源テーブル管理部 15 A には、管理テーブル 18 A を設ける。管理テーブル 18 A は、コピーによって作成された資源テーブル管理部 35 A から参照可能なメモリ領域である。管理テーブル 18 A は、環境マネージャ 11 A に従属するすべての環境マネージャ (環境マネージャ 31 A を含む) のアドレスが格納される。管理テーブル 18 A は、環境マネージャ 11 A が初めてコピーを行う時に、資源テーブル管理部 15 A によって作成されてもいいし、あるいは始めから資源テーブル管理部 15 A に設けられていてもよい。

【0115】ここで、資源テーブル管理部 15 A に、次の手続きを設ける。

【0116】(g) (f2) により、資源テーブル管理部 35 A から送られた SUBEMS 通知 705 に基づき、資源テーブル管理部 35 A のアドレスを管理テーブル 18 A に追加する。

【0117】(h) 手続き (d) (e) に引き続き、管理テーブル 18 A を参照し、そこに記載の環境マネージャの資源テーブル管理部 35 A に環境マネージャ 11 B ~ 11 N に送ったのと同じ NEWLOAD 通知 801 を送信する。

【0118】資源テーブル管理部 35 A は、手続き (d) (e) その他の、資源テーブル管理部 15 A が有する手続きを持っている。また、資源利用制御部 34 A と同じく、資源テーブル 33 A を参照することができる。この資源テーブル管理部 35 A に、次の手続きを設ける。

【0119】(g2) 資源テーブル管理部 35 A は、資源テーブル 13 A をその一部として含んでいる資源テーブル 33 A を作成する。

【0120】(h2) 資源テーブル管理部 35 A は、手続き (h) において、資源テーブル管理部 15 A より送られた NEWLOAD 通知 801 を受信したら、資源テーブル 33 A の各エントリ 130 に記載の環境マネージャのアドレス 131 と、受信した NEWLOAD 通知のデータ 203 に記載のエントリ 130 に記載の環境マネージャのアドレス 131 を比較する。一致するエント

リ130を見出し、もしエントリの内容が異なっている時は、NEWLOAD通知のデータ203に記載のエントリ130に置き換える。さらに、もし置き換えられた資源テーブル33Aのエントリが、資源テーブル13Aのエントリである時は、環境マネージャのアドレス資源テーブル33Aの、資源テーブル13A以外の各エントリを参照し、そのエントリに記載されているアドレスの環境マネージャの資源テーブル管理部(図8では25A~25N)に対して、手続き(d)と同じ手段でSETLOAD通知802を送信し、資源テーブル13Aに記載のプロセッサの負荷に変化が生じたことを通知する。

【0121】以上の手続きを設けることにより、資源テーブル33A、資源利用制御部34A、資源テーブル管理部35Aからなる環境マネージャ31Aが作成される。

【0122】アプリケーション1は、環境マネージャ31Aを通じて、これまで環境マネージャ11Aを通じて行っていたのと同様に、資源テーブル33Aに記載の分散計算機資源(プロセッサ)を利用することができる。

【0123】すなわち、アプリケーション1のアダプタ7から送られたSYSCALL通知は資源利用制御部34Aにおいて受信され、資源利用制御部34Aは手続き(a)を呼び出して、資源テーブル33Aに記載のエントリの中から、最適な(負荷の小さい)計算機を選択し、すでに述べたREXEC通知をそのエントリに記載の環境マネージャに送信し、その計算機上でアプリケーションを新規に実行する。

【0124】また、REXEC通知の送信の後、資源利用制御部34Aの手続き(c)が引き続き呼び出され、資源テーブル33Aに記載の計算機の中の1つの負荷が変化したことを知らせるため、NEWLOAD通知が送信される。ここで、資源利用制御部34Aの手続き

(c)は、資源利用制御部14Aの手続き(c)と同様、資源テーブル管理部15Aの方にNEWLOAD通知を送る(804)。

【0125】また、別の計算機からのREXEC通知は、環境マネージャ11Aをコピーする前と同様、資源利用制御部14Aによって受信され、資源利用制御部14Aの手続き(b)が呼び出されて処理され、資源利用制御部14Aの手続き(c)によりNEWLOAD通知が資源管理テーブル15Aに送信される(404)。

【0126】上で設けた手続き(h)(h2)は、環境マネージャをコピーを行った後もコピーをする前と同様に、1つの計算機上での負荷の変化を、関連するすべての環境マネージャにSETLOAD通知により知らせ、それぞれの環境マネージャの資源テーブルの状態を更新させるための手続きである。

【0127】資源テーブル13Aに記載された環境マネージャ(より正確には、環境マネージャの資源テーブル

管理部)へのSETLOAD通知の送信は、環境マネージャのコピーの前と同様に、資源テーブル管理部15Aによって行われる。また、資源テーブル33Aに記載の環境マネージャの内、資源テーブル13Aに記載されたもの以外の環境マネージャへのSETLOAD通知の送信は、資源テーブル管理部35Aにおいて、手続き(h2)により行われる。

【0128】一方、SETLOAD通知の受信は、環境マネージャのコピー前と同様に、資源テーブル管理部15Aの手続き(e)によって行われる。環境マネージャ11Aに従属する環境マネージャが存在する場合(これは管理テーブル18Aによって調べることができる)、手続き(e)に引き続いて手続き(h)が呼び出される。手続き(h)は、NEWLOAD通知801を資源テーブル管理部35Aに送信し、資源テーブル33Aを更新させる。

【0129】(8)資源テーブルの拡張

本実施形態では、アプリケーションが使用する環境は環境マネージャの資源テーブルの内容により定まるので、環境の拡張は資源テーブルの内容を追加することによって、アプリケーションが利用できる環境を拡張することができる。資源テーブルの内容を追加する具体的な方法は、例えば、図1においてアプリケーション1が既に計算機10A~10Nのプロセッサを使用可能な状態で更に環境を拡張しようとするれば、環境マネージャ11Aの資源テーブル13Aのエントリに、プロセッサ16A~16N以外のプロセッサに対応するエントリを追加することである。この結果、アプリケーション1が利用可能なプロセッサの数が増加し、負荷分散によりアプリケーションの実行速度が高まる可能性が大きくなる。

【0130】この資源テーブルを拡張する手続きを図9および図1を用いて説明する。図9は、環境マネージャがコピーされる場合での資源テーブルの拡張を説明する場合の計算機の内部機能を示すものである。図1における計算機のそれぞれが、この内部機能を持つことになる。ここで、アプリケーション1は、計算機10A~10N上の環境マネージャ11A~1Nにより、プロセッサ16~16Nを利用可能になっているものとする。また、アプリケーション2は、計算機20A~20N上の環境マネージャ21A~21Nにより、プロセッサ26A~26Nを利用可能になっているものとする。

【0131】まず、資源利用制御部14Aに、次の手続きを設ける。

【0132】(i) アプリケーション1がアプリケーション2との接続を指定した「接続要求」要求を検出し(901)、接続1000が確立されたら、手続き(f)を実行し、環境マネージャ31Aを作成、実行する。

【0133】(j) アプリケーション1の「接続受付」要求を検出し(901)、接続が確立されたら、手

続き(f)を実行し、環境マネージャ31Aを作成、実行する。

【0134】また資源利用制御部34Aには、次の手続きを設ける。

【0135】(i2) 手続き(f2)の後、アプリケーション1とアプリケーション2が通信を始める前に、両アプリケーション間に確立された接続を通して、資源テーブル33Aの内容をFUSION通知902のデータ203として送信する。また、この接続から受信したFUSION通知902を、資源テーブル管理部35A 10 に送信する(903)。

【0136】更に、資源テーブル管理部35Aには、次の機能を設ける。

【0137】(k) 手続き(i)(j)により、資源利用制御部34Aから送られてきた内容を資源テーブル33Aへの追加分53Aとして、資源テーブル33Aを更新する。

【0138】以上に示した手続きを、環境マネージャ11Aおよび21Aに設けることにより、アプリケーション1は、アプリケーション2との通信を契機として、アプリケーション2が利用可能なプロセッサ26A~26Nを新たに利用することができるようになる。 20

【0139】より詳細に述べると、アプリケーション1がアプリケーション2と通信しようとする場合、アプリケーション1はアプリケーション2に対して接続要求を行なう。ここで、アプリケーション2が接続受付を行なっていれば、アプリケーション1とアプリケーション2の間で1つの接続が確立される。この際、アプリケーション1とアプリケーション2とが確立された接続を通じて通信を始める前に、環境マネージャ11Aでは手続き(i)、環境マネージャ21Aでは手続き(j)により、環境マネージャ11Aのコピー31Aおよび環境マネージャ21Aのコピー(これを環境マネージャ41Aとする)が作成、実行される。

【0140】環境マネージャ31Aには、手続き(i2)により、両アプリケーション間で確立された接続を通して、資源利用制御部34Aに資源テーブル23Aの内容が送られ、資源利用制御部34Aはそれを資源テーブル管理部35Aに送る。資源テーブル管理部35Aは、手続き(k)により、受けとった資源テーブル23Aの内容を資源テーブル33Aに追加する。これにより、資源テーブル33Aはプロセッサ26A~26Nの分だけそのエントリを増やし、アプリケーション1は、以後環境マネージャ31Aを通じ、既に述べた手続き(a)~手続き(e)に基づき、それまで利用可能だったプロセッサ16A~16Nに加え、プロセッサ26A~26Nが利用可能となる。

【0141】逆に、アプリケーション1が「接続受付」要求を行なっており、その後アプリケーション2からの接続要求がアプリケーション1に届いた場合は、環境マ 50

ネージャ11Aでは手続き(j)、環境マネージャ21Aでは手続き(i)が呼び出され、同様にプロセッサ26A~26Nが新たに利用可能となる。

【0142】環境マネージャ21Aのコピーである環境マネージャに関しても環境マネージャ31Aと同様なことが言える。

【0143】手続き(i)および手続き(j)において、コピーである環境マネージャ31Aと環境マネージャ21Aのコピーである環境マネージャ(これを環境マネージャ41Aとする)が通信する時に、アプリケーション1とアプリケーション2との間で確立された接続を利用しているが、これには次のような利点がある。

【0144】両アプリケーションの通信を契機に、相手のアプリケーションが利用可能なプロセッサを追加する形で、自分が利用可能なプロセッサを拡張する場合、本実施形態では環境マネージャ31Aと環境マネージャ41Aとの間で資源テーブルを送受信する手続きが必要である。しかし、資源テーブル13Aには(従って資源テーブル33Aにも)環境マネージャ21A(あるいは41A)への通信方法が記載されていないし、資源テーブル23A(あるいは43A)についても同様である。しかし、アプリケーション1とアプリケーション2とが通信できるとすると、両アプリケーションの少なくとも一方は、相手のアプリケーションに対する通信方法を知っている、あるいは知る方法が存在するのが普通である。そこで、手続き(i)、手続き(j)では、両アプリケーション間の接続が確立されたあと、アプリケーションにその旨を通知する前に、環境マネージャ31Aと41Aが、その確立された接続を通して、互いの資源テーブルを送り合う。このため、環境マネージャ11A及び21Aは、アプリケーション1~2の通信を契機として資源テーブルを拡張する際に、相手に対する通信方法をあらかじめ知っている必要がないという利点がある。

【0145】以上述べたように、本実施形態では、アプリケーション1とアプリケーション2との通信開始を契機に、まず環境マネージャのコピーを行ない、その後コピーした環境マネージャ31Aと41Aが、自分の資源テーブル33Aと43Aを互いに送信し合い、受信した資源テーブルを自分の資源テーブルに追加する構造となっている。これにより、アプリケーション1とアプリケーション2のみが相手の分散計算機資源を利用でき、また、アプリケーション1とアプリケーション2の協調動作が終了し次第、環境の拡張を解消することができる。このため、本実施形態では、あるアプリケーションが、他のアプリケーションが利用可能な分散計算機資源を、不当に利用してしまうのを防ぐことができる。

【0146】(9) 通信終了後の資源テーブルの復旧  
アプリケーション1とアプリケーション2との通信を契機として、自分の利用可能な計算機資源を相手にも利用できるようにする形で、お互いの環境を拡張した場合、

両アプリケーションの協調動作が終了した時点で、相手が利用可能な計算機資源を、自分はそれ以上使えなくなるのが実用上望ましい。この機能を実現するため、本実施形態では、環境マネージャ 31A の資源利用制御部 34A に次の手続きを設ける。なお、ここでの説明では環境マネージャのコピー後の状態を示す図 7 を参照するが、これら復旧に関する手続き自体は図示していない。

【0147】(m2) アプリケーション 1 から OS 3 への「閉鎖」または「遮断」要求の発行を検出すると、アダプタ 7 に SUBEME 通知を送信して、以後は環境マネージャ 11A の資源利用制御部 14A に SYSCALL 通知を送るよう通知する。次に、資源テーブル管理部 15A に SUBEME 通知を送信して、環境マネージャ 31A が終了する旨を通知する。その後、資源テーブル管理部 35A を終了させ、自分自身も終了する。

【0148】一方、環境マネージャ 11A の資源利用制御部 14A には、次の手続きを設ける。

【0149】(m) 資源利用制御部 34A からの SUBEME 通知を受けて、管理テーブル 18A から環境マネージャ 31A に対応するエントリを削除する。

【0150】これにより、アプリケーション 1 は、アプリケーション 2 との通信が終了した後、環境マネージャ 11A を通じて、拡張する以前に利用可能であった分散計算機資源を利用しつつ、実行を続けることができる。例えば、この後で、アプリケーション 1 が、アプリケーション 2 とはまた別のアプリケーションと通信する場合でも、これまで述べてきた手続きの繰り返しにより、同様にそのアプリケーションと利用可能な分散計算機資源を提供しあうことができる。

【0151】

【発明の効果】本発明では、新たに通信を開始する 2 つのアプリケーションが、それまで自分が利用可能であった分散計算機資源を、通信相手のアプリケーションも利用できるようにすることができる。これにより、アプリケーションは、より多くの分散計算機資源を有効利用できるようになり、アプリケーションの実行性能を高めることができる。特に、アプリケーション同士が広域ネットワークを経由して通信する必要がある時、利用可能な分散計算機資源を大きく拡張することができる。

【図面の簡単な説明】

【図 1】本発明の実施形態の全体像となる、ネットワークに接続された計算機と分散計算機資源から成る、分散プログラミング環境を示す図である。

【図 2】従来の技術において、アプリケーションから分散計算機資源が利用可能となっている様子を示す図である。

【図 3】本発明の実施形態において、資源テーブルの内容を示す図である。

【図 4】本発明の実施形態において、アプリケーションから分散計算機資源の利用を可能にする手続きを示す図

である。

【図 5】本発明の実施形態において、アダプタと環境マネージャの関係を示す図である。

【図 6】本発明の実施形態における、通知の形式を示す図である。

【図 7】本発明の実施形態における、環境マネージャのコピーを実現する手続きを示す図である。

【図 8】本発明の実施形態における、環境マネージャをコピーした後で、アプリケーションからの分散計算機資源の利用を実現する手続きを示す図である。

【図 9】本発明の実施形態における、資源テーブルの拡張を実現する手続きを示す図である。

【図 10】本発明の実施形態の適用対象例を示す図である。

【図 11】図 10 において、本発明の実施形態の動作の概略における留意点を示す図である。

【図 12】図 10 において、本発明の実施形態の動作の概略を示す図である。

【図 13】本発明の実施形態において、環境の設定を行う手続きを示す図である。

【図 14】通知の内容を示す図である。

【符号の説明】

1 アプリケーション

2 別のアプリケーション

7 アダプタ

10A~10N 計算機

11A~11N 環境マネージャ (アプリケーション 1 の環境を管理)

13A~13N 資源テーブル

14A~14N 資源利用制御部

15A~15N 資源テーブル管理部

16A~16N アプリケーション 1 が利用可能な分散計算機資源 (プロセッサ)

18A 管理テーブル

20A~20N 計算機

21A~21N 環境マネージャ (アプリケーション 2 の環境を管理)

23A~23N 資源テーブル

24A~24N 資源利用制御部

25A~25N 資源テーブル管理部

26A~26N アプリケーション 2 が利用可能な分散計算機資源 (プロセッサ)

31A 環境マネージャ 11A に従属する環境マネージャ

41A 環境マネージャ 21A に従属する環境マネージャ

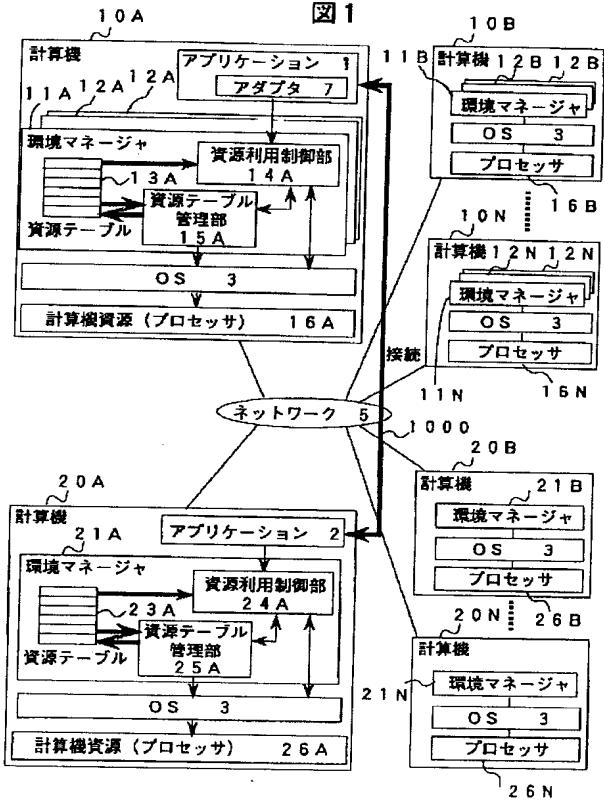
60A 環境の設定時に、追加すべき計算機資源を有する計算機

61A 環境の設定時に、追加すべき計算機資源を管理する環境マネージャ



- 31
- 130 資源テーブルのエントリ
  - 131 環境マネージャのアドレス
  - 132 プロセッサの識別番号

【図1】



【図3】

図3

13A

131	132	133	
11Aのアドレス	16Aの種類	16Aの負荷	130
11Bのアドレス	16Bの種類	16Bの負荷	130
131	132	133	...
131	132	133	...
11Nのアドレス	16Nの種類	16Nの負荷	130

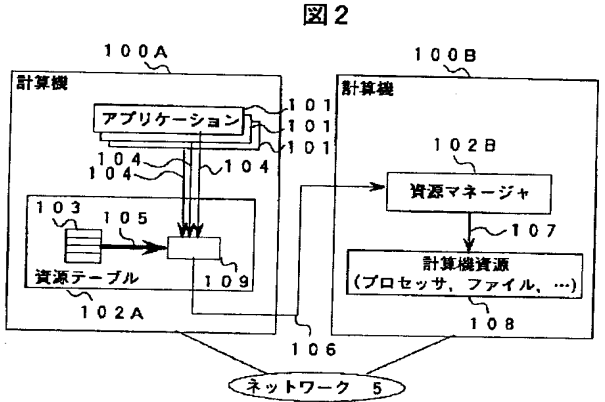
資源テーブル

14Nのアドレス	15Nのアドレス
----------	----------

131

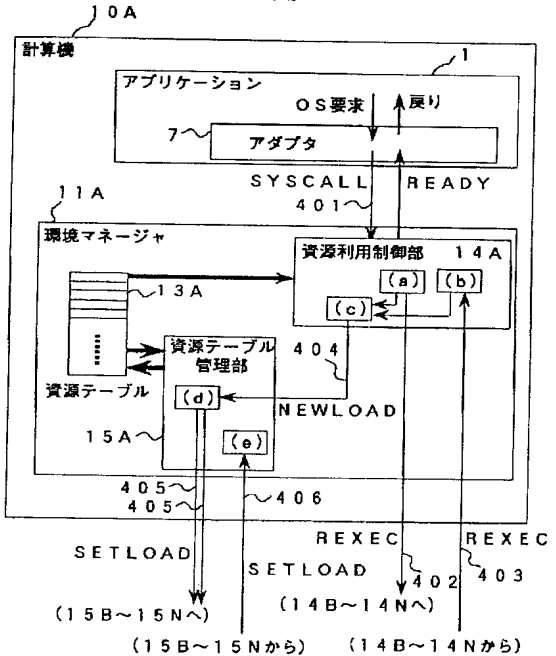
- 133 プロセッサの負荷
- 1000 アプリケーション1とアプリケーション2との通信開始（接続）。

【図2】



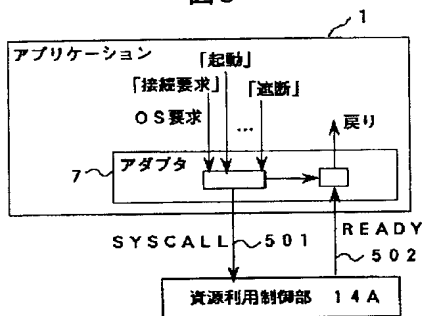
【図4】

図4



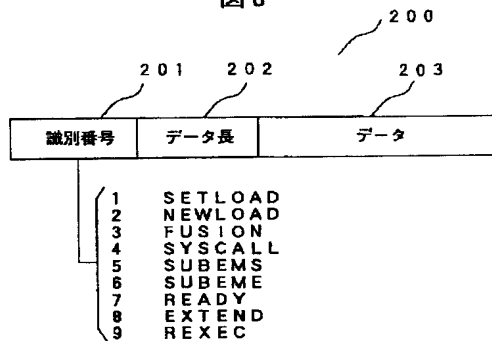
【図5】

図5



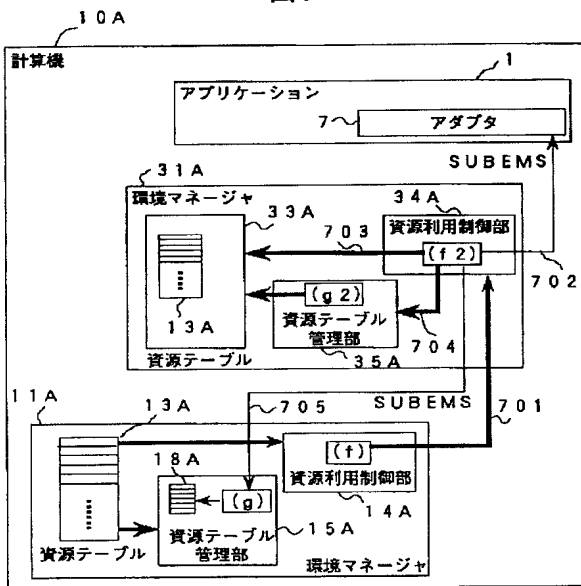
【図6】

図6



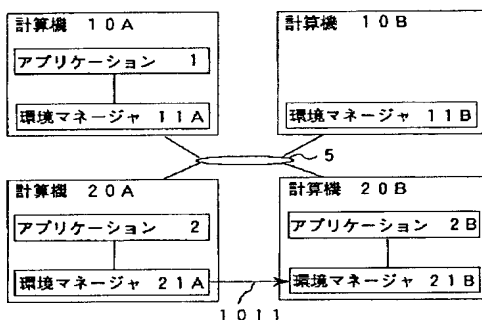
【図7】

図7



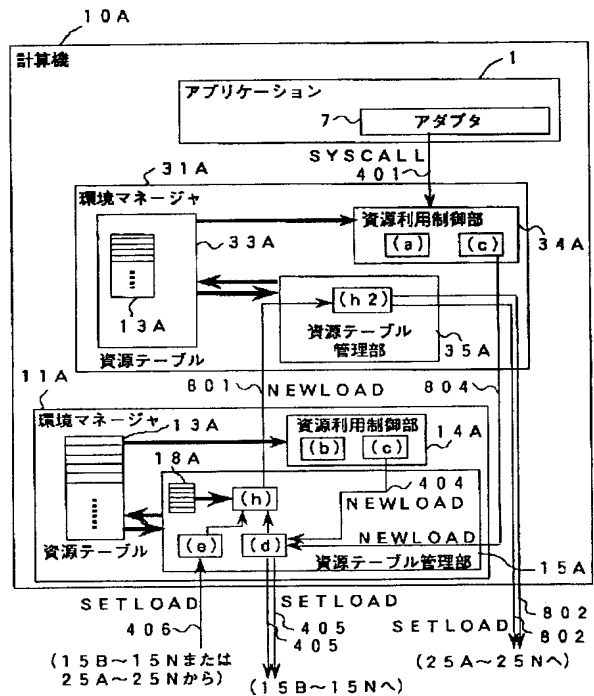
【図10】

図10



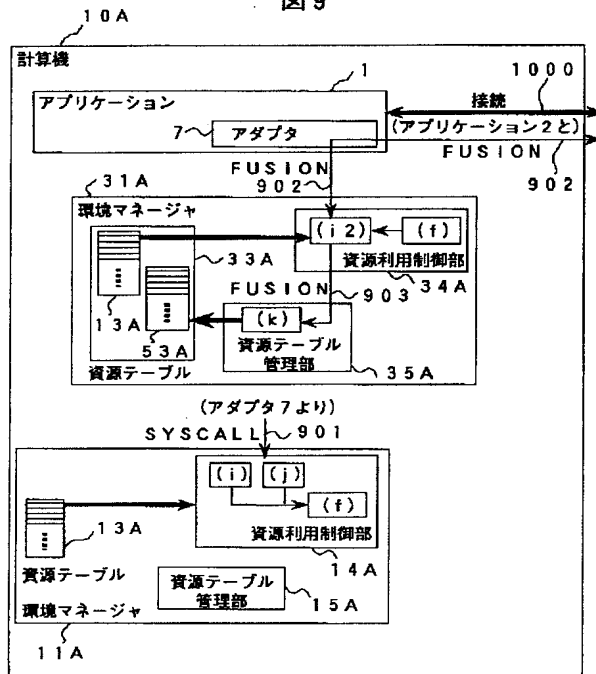
【図8】

図8



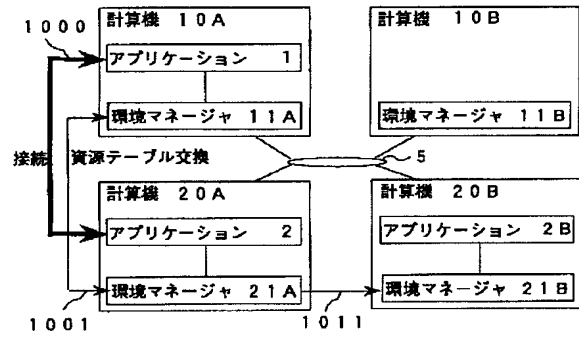
【図9】

図9



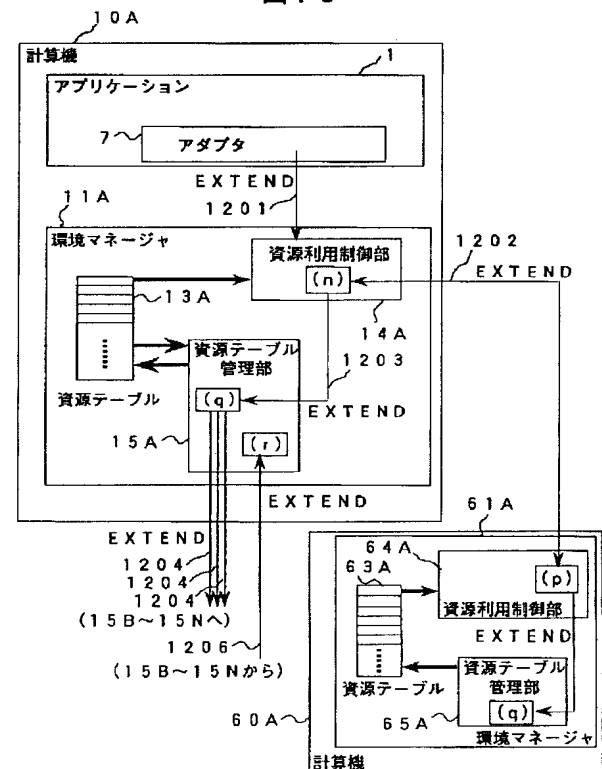
【図11】

図11



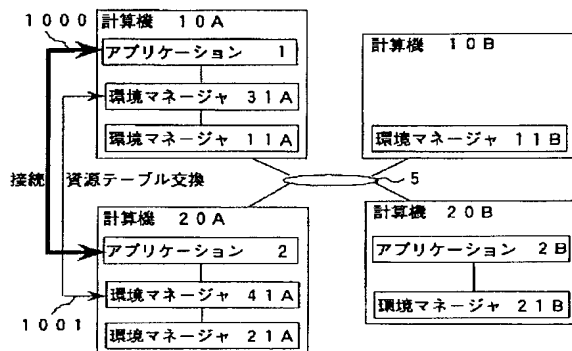
【図13】

図13



【図12】

図12



【図 14】

図 14

識別番号	通知名	データの内容
1	SETLOAD	エントリ
2	NEWLOAD	エントリ
3	FUSION	資源テーブルの内容
4	SYSCALL	OSへの要求の種類、および引数
5	SUBEMS	環境マネージャのアドレス
6	SUBEME	環境マネージャのアドレス
7	READY	環境マネージャのアドレス
8	EXTEND	環境マネージャのアドレス、または 環境マネージャの資源テーブルの内容
9	REXEC	実行すべきアプリケーション

---

フロントページの続き

(72)発明者 野田 文雄

東京都国分寺市東恋ヶ窪 1 丁目280番地  
株式会社日立製作所中央研究所内

## PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-244940

(43)Date of publication of application : 19.09.1997

(51)Int.Cl.

G06F 12/00

G06F 12/00

G06F 15/16

(21)Application number : 08-054398

(71)Applicant : HITACHI LTD

(22)Date of filing : 12.03.1996

(72)Inventor : MASUOKA YOSHIMASA  
KAGIMASA TOYOHICO  
KITAI KATSUYOSHI  
NODA FUMIO

## (54) METHOD FOR MANAGING DISTRIBUTED COMPUTER RESOURCE

## (57)Abstract:

**PROBLEM TO BE SOLVED:** To use distributed computer resources for an application from a different application when the application starts communication with the different application.

**SOLUTION:** Resource tables 13A and 23A for managing the computer resources utilizable by the applications 1 and 2 are provided corresponding to the applications 1 and 2. When a certain application 1 is connected to the different application 2 and performs a cooperative processing, by preparing the resource tables 13A and 23A for managing the resources of the both by mutually copying the respective resource tables 13A and 23A and referring to them by the respective applications 1 and 2, the respective applications 1 and 2 utilize the computer resources of the both. Thus, the two applications mutually utilize the distributed computer resources utilized by them until then, many more distributed computer resources are effectively utilized and execution performance is improved.

